How to Guarantee Secrecy for Cryptographic Protocols

Danièle Beauquier and Frédéric Gauche

LACL CNRS FRE 2673, Université Paris 12 Val de Marne
61 Avenue du Général de Gaulle F-94010 Créteil Cedex, France
{beauquier, gauche}@univ-paris12.fr

Abstract. In this paper we propose a general definition of secrecy for cryptographic protocols in the Dolev-Yao model. We give a sufficient condition ensuring secrecy for protocols where rules have encryption depth at most two, that is satisfied by almost all practical protocols. The only allowed primitives in the class of protocols we consider are pairing and encryption with atomic keys. Moreover, we describe an algorithm of practical interest which transforms a cryptographic protocol into a secure one from the point of view of secrecy, without changing its original goal with respect to secrecy of nonces and keys, provided the protocol satisfies some conditions. These conditions are not very restrictive and are satisfied for most practical protocols.

Keywords: Security protocols, secrecy, cryptographic protocols.

1 Introduction

Cryptographic protocols are used to ensure secure communications between two or more parties in a distributed system. Among the requirements that cryptographic protocols must satisfy are the well-known authentication and secrecy or confidentiality.

Security protocol design and verification is a very hard problem. Sources of difficulty are numerous and of different types. The seminal paper for developing a model was proposed by D. Dolev and A.C. Yao. [DY83]. In recent years a lot of methods have been proposed for reasoning about cryptographic protocols. Some of them are based on the trace model [Pau98,JG02] including models with an explicit state-transition system [CDL$^+$99] or Horn clauses [Bla01,CCM01]. Another type of model uses processes to represent cryptographic protocols [AG00,Sch97].

Concerning secrecy there are basically two approaches, the first one reduces the secrecy property to a reachability problem, the second one defines secrecy in terms of an observability equivalence.

Most of the papers are devoted to decidability and undecidability results depending on various hypothesis related to the boundedness of nonces and sessions, the used cryptographic primitives and so on. See for example [DLMS04] for a review of these results. Surprisingly, there are very few results that give

some rules to apply in order to guarantee the secrecy property. This question has already been answered in the case of cryptographic protocols using symmetric keys in [Bea04], which gives a sufficient condition for solving this problem. Here we consider a more general class of cryptographic protocols using both symmetric and asymmetric keys. We give a new sufficient condition adapted to this type of protocols and we describe an algorithm of practical interest. It transforms a cryptographic protocol (satisfying a condition which is not very restrictive) into a secure one from the point of view of secrecy, without changing its original goal with respect to secrecy of nonces and keys.
In Section 2 we describe the model. Section 3 gives the sufficient condition for a secure protocol w.r.t secrecy, Section 4 is devoted to the algorithm which transforms a cryptographic protocol into a secure one w.r.t secrecy. The last section concludes.
Related work As mentioned before, the security literature concentrates more on the verification of cryptographic protocols comparing to the synthesis of correct protocols. In [AN96], some prudent principles for designing protocols are given but do not guarantee the success. Several of these principles are present in our definition of well composed protocol. A sufficient condition based on typing is presented in [Aba99] but it concerns only symmetric keys and a binary view of secrecy according to which the world is divided into system and attacker. Our sufficient condition can be considered as a generalization of the sufficient condition given in [Low95]. Indeed, the protocols which are considered in this paper do not admit forwarding, which is an important restriction. At last, to our knowledge, there is no paper which describes an algorithm which transforms a protocol into a secure one w.r.t. secrecy and preserves its original goal.

2 The model

In this section we formalize the model we use and we specify the assumptions we make about protocols commonly referred to as the "Dolev-Yao model". Our approach is largely inspired by [Low99]. The only primitives used are pairing and encryption. We assume that pairing is *associative*, which corresponds to practical protocols, so the algebra of terms is the quotient of a free algebra with equations for associativity.

We are interested in the behavior of the protocol when the number of agents, nonces and sessions is *unbounded*. Moreover the hypothesis on the power of honest agents is as weak as possible. The knowledge of an agent is *local*, it does not have a global memory of all its sessions. On the contrary the power of the intruder is maximal.

2.1 Messages

Atomic values The set *Value* is a set of disjoint types *Agent*, *Nonce*, *Key*, *Cypher*. *Agent*, *Nonce*, *Key* are sets of atomic values. *Cypher* is the set of

values obtained by encryption. The set *Agent* is the set of all agent identities. It is partitioned into two subsets, honest agents and intruders: $Agent = Honest \cup Dishonest$. W.l.o.g. one supposes that the set *Dishonest* contains a unique intruder $\mathcal{I}$. Agents variables, named A, B, ..., belong to the set *AgVar*. *Nonce* is an infinite set of integers. Nonce variables named *N*, *Na*, *Nb*, ... , belong to the set *NVar*.
The set *Key* is divided into two disjoint subsets *ShKey* and *LKey*.
- *ShKey* is the set of *short term keys*. Its elements correspond to symmetric keys used only for the current session.
- *LKey* is the set of *long term keys*. It is a disjoint union of *SymKey* the set of symmetric keys and *AsymKey* the set of asymmetric keys.
The set *AsymKey* is a disjoint union of two subsets *PubKey* (public keys) and *PrivKey* (private keys).
Short term key variables named $\mathcal{K}$, $\mathcal{K}'$, ... belong to the set *KVar*.
Let A, B be two agent variables. We denote respectively by $\mathcal{K}_{priv}(A)$, $\mathcal{K}_{pub}(A)$, $\mathcal{K}(A, B)$, the long term private encryption key of A, the long term public encryption key of A, the long term symmetric key shared by agents A and B. Notice that in this notation $\mathcal{K}_{priv}(A)$, $\mathcal{K}_{pub}(A)$ are both encryption keys, and $\mathcal{K}_{pub}(A)$ is NOT the inverse key of $\mathcal{K}_{priv}(A)$ and vice-versa.

Symbolic terms. *Symbolic terms* are constructed using *pairing* and *encryption.*
The pairing of terms $\mathcal{X}$ and $\mathcal{Y}$ is the term $< \mathcal{X}, \mathcal{Y} >$, the encryption of term $\mathcal{X}$ using the key $\mathcal{K}$ is $\{\mathcal{X}\}_{\mathcal{K}}$, C is a set of constants.
The grammar used to generate symbolic terms is :
$key ::= C \mid KVar \mid \mathcal{K}_{pub}(AgVar) \mid \mathcal{K}_{priv}(AgVar) \mid \mathcal{K}(AgVar, AgVar)$
$symb\_term ::= NVar \mid AgVar \mid key \mid < symb\_term, symb\_term > \mid \{symb\_term\}_{key}$
The pairing is associative, or equivalently for each n we have a primitive for n-pairing. We will consider only terms which are in a "canonical form". For example the canonical form of terms $< t_1, < t_2, t_3 >>$ and $<< t_1, t_2 >, t_3 >$ is $< t_1, t_2, t_3 >$, it means that $< t_1, < t_2, t_3 >>$ and $< t_1, t_2, t_3 >$ must be considered as triples and not pairs.

The set of subterms of a term τ is denoted $Sub(\tau)$.

Concrete terms. *Concrete terms* are generated following the same grammar as for symbolic terms, except that variables in $NVar, KVar, AgVar$ are replaced by the values of the corresponding type.
$Key ::= ShKey \mid \mathcal{K}_{pub}(Agent) \mid \mathcal{K}_{priv}(Agent) \mid \mathcal{K}(Agent, Agent)$
$conc\_term ::= Nonce \mid Agent \mid Key \mid < conc\_term, conc\_term > \mid \{conc\_term\}_{Key}$

Synthesis and Analysis. In this subsection, 'term' means 'symbolic term'. The synthesis procedure represents terms the agents can build. The analysis procedure represents terms the agents can learn.

Let $\mathcal{T}$ be a set of terms and A be an agent variable. The set $Synth_A(\mathcal{T})$ is the least set of terms containing $\mathcal{T}$ and satisfying:

– $\tau_1, ..., \tau_p \in Synth_A(\mathcal{T}) \Rightarrow < \tau_1, ..., \tau_p > \in Synth_A(\mathcal{T})$ (An agent can compose the terms he knows).
– $\forall \tau \in Synth_A(\mathcal{T}), \forall B \in AgVar$
  - $\{\tau\}_{\mathcal{K}(A,B)} \in Synth_A(\mathcal{T})$ (An agent can encrypt with a symmetric key he shares with another agent)
  - $\{\tau\}_{\mathcal{K}_{priv}(A)} \in Synth_A(\mathcal{T})$ (An agent can encrypt with his own private key).
  - $\{\tau\}_{\mathcal{K}_{pub}(B)} \in Synth_A(\mathcal{T})$ (An agent can encrypt with the public key of any agent).
– $\forall \tau \in Synth_A(\mathcal{T})$ and for all short term key variable $\mathcal{K} \in \mathcal{T}$, $\{\tau\}_{\mathcal{K}} \in Synth_A(\mathcal{T})$ (An agent can encrypt with all encrypting short term key he knows).

Let A be an agent variable. Let $\mathcal{T}$ and $\mathcal{T}'$ be two sets of terms.
We have $\mathcal{T} Anal_A \mathcal{T}'$ if one of the following properties holds:

– $< \tau_1, ..., \tau_p > \in \mathcal{T}, p > 1$ and $\mathcal{T}' = (\mathcal{T} \setminus \{< \tau_1, ..., \tau_p >\}) \cup \{\tau_1\} \cup ... \cup \{\tau_p\}$ (An agent can decompose terms).
– $\{\tau\}_{\mathcal{K}} \in \mathcal{T}$, $\mathcal{K} \in \mathcal{T}$ is a symmetric session key variable, and $\mathcal{T}' = (\mathcal{T} \setminus \{\tau\}_{\mathcal{K}}) \cup \{\tau\}$ (An agent can decrypt terms encrypted with a short term session key he knows).
– $\{\tau\}_{\mathcal{K}(A,B)} \in \mathcal{T}$, $B \in \mathcal{T}$ and $\mathcal{T}' = (\mathcal{T} \setminus \{\tau\}_{\mathcal{K}(A,B)}) \cup \{\tau\}$ (An agent can decrypt terms encrypted with a key shared with an agent he knows.)
– $\{\tau\}_{\mathcal{K}_{pub}(A)} \in \mathcal{T}$ and $\mathcal{T}' = (\mathcal{T} \setminus \{\tau\}_{\mathcal{K}_{pub}(A)}) \cup \{\tau\}$ (An agent can decrypt terms encrypted with his own public key).
– $\{\tau\}_{\mathcal{K}_{priv}(B)} \in \mathcal{T}$, $B \in \mathcal{T}$ and $\mathcal{T}' = (\mathcal{T} \setminus \{\tau\}_{\mathcal{K}_{priv}(B)}) \cup \{\tau\}$ (An agent can decrypt terms encrypted with the private key of an agent he knows).

A set of terms $\mathcal{T}$ is told *undecomposable* if there does not exist any set of term $\mathcal{T}'$ such that $\mathcal{T} Anal_A \mathcal{T}'$ (An agent cannot decompose any more term).
It is easy to prove that for any set of terms $\mathcal{T}$, there exists a unique undecomposable set of terms $\mathcal{T}'$ such that $\mathcal{T} Anal_A^* \mathcal{T}'$. This set is denoted $Anal_A^*(\mathcal{T})$.

For a term $\tau \in Anal_A^*(\mathcal{T})$ we define *the number of steps necessary for A to learn τ from $\mathcal{T}$* as the number of decryption operations that A must use before obtaining τ, more precisely :

– A learns τ from $\mathcal{T}$ in 0 step iff some term $< ..., \tau, ... >$ is in $\mathcal{T}$ (we admit here a composition of a single element τ). (No decryption necessary).
– if A learns $\{< ..., \tau, ... >\}_{\mathcal{K}}$ from $\mathcal{T}$ in at most p steps and $\mathcal{K}$ is a short session symmetric key learnt by A from $\mathcal{T}$ in at most q steps, then τ is learnt by A from $\mathcal{T}$ in at most $p + q + 1$ steps.
– if A learns $\{< ..., \tau, ... >\}_{\mathcal{K}(A,B)}$ from $\mathcal{T}$ in at most p steps and B is learnt by A from $\mathcal{T}$ in at most q steps, then τ is learnt by A from $\mathcal{T}$ in at most $p + q + 1$ steps.

– if A learns $\{< ..., \tau, ... >\}_{\mathcal{K}_{pub}(A)}$ from $\mathcal{T}$ in at most p steps then τ is learnt by A from $\mathcal{T}$ in at most $p+1$ steps.
– if A learns $\{< ..., \tau, ... >\}_{\mathcal{K}_{priv}(B)}$ from $\mathcal{T}$ in at most p steps and B is learnt by A from $\mathcal{T}$ in at most q steps, then τ is learnt by A from $\mathcal{T}$ in at most $p+q+1$ steps.

For $\tau, \tau' \in Anal_A^*(\mathcal{T})$ we define that A learns τ from $\mathcal{T}$ before τ' if A learns τ in p steps, τ' in p' steps and $p < p'$.
Now, given a concrete agent a, we can define in the same way a relation $Anal_a$ on finite sets of concrete terms as well as the other notions defined above, replacing agents, and keys variables by values of the corresponding type.

Message (Component, Protocol) -Template A *component template* is either a variable or an encrypted term. A *message template* or *t-message* is a tuple of the form (A, B, τ) where A and B are distinct variables of agents representing respectively the sender and the receiver and τ is a term representing the *content* of the message.
A *concrete message* is a tuple (a, b, m) where a and b are agent values and m is a concrete term. It corresponds to the informal usual notation $A \rightarrow B : m$.
A *protocol template* or simply *protocol* is a sequence of message templates.
A *role* in a protocol template is an agent variable appearing in this protocol.
Given a protocol P with a set of roles $\mathcal{R}$, a *session template* Ses_A for role $A \in \mathcal{R}$ is the subsequence of message templates of P in which role A is sender or receiver.
Our running example will be the protocol TMN [TMN90] using asymmetric keys. Brackets for pairing are omitted as usual.

Example 1. .
$01 - A \rightarrow S : B, \{K_a\}_{\mathcal{K}_{pub}(S)}$
$02 - S \rightarrow B : B, A$
$03 - B \rightarrow S : A, \{K_b\}_{\mathcal{K}_{pub}(S)}$
$04 - S \rightarrow A : B, \{K_b\}_{K_a}$

Ses_S is the entire protocol. Ses_A is the sequence:
$A \rightarrow S : B, \{K_a\}_{\mathcal{K}_{pub}(S)}$
$S \rightarrow A : B, \{K_b\}_{K_a}$

2.2 Realizable protocol template

An elementary question is whether a protocol is “realizable”, i.e. whether the honest agents can execute it. This notion appears in [RS03] as ”well-formed” protocol. We formalize this notion in our framework and give an algorithm which checks whether a protocol is realizable or not. One can observe that as far as we are aware of, most of the undecidability proofs [DLMS99,AC02a,AC02b] are based on protocols which are not realizable, which is a weakness of these proofs. Only in [CCM01] the undecidability proof relies on realizable protocols.
Let P be a protocol, and A be a role of this protocol. Consider the sequence of

t-messages of the session template Ses_A. The j^{th} t-message of Ses_A is of form (A, B_j, τ_j) or (B_j, A, τ_j) depending on A is sender or receiver of the message.
We define $Kn_{A,j}$ as the *knowledge* of role A after execution of message number j. That is to say as the set of terms known by A after the execution of the first j t-messages of his session and that A can no more decompose.
This knowledge can be decomposed into two subsets:

- The *basic knowledge* of A at step j, $BasKn_{A,j}$, which contains agent, nonce and key variables.
- The *cryptographic knowledge* of A at step j, $CrKn_{A,j}$, which contains the encrypted terms known by A at step j and he cannot decrypt.

Notice that $Kn_{A,j}$ contains only terms which are component templates.
From the definition of synthesis, we can define $Synth_A(Kn_{A,j})$ as the set of terms that A can build from his knowledge at step j.
Let us define by induction on j the set $Kn_{A,j}$ and the fact that the j first messages of Ses_A are realizable.
The *initial knowledge* of A, $Kn_{A,0}$ is fixed by the protocol.
We need to introduce the notion of new variables appearing in a t-message of protocol P. Let (A_p, B_p, τ_p) be the p^{th} t-message of P. The set of new variables of this t-message denoted $NewVar_p$ is defined recursively:
$NewVar_1 = Sub(\tau_1) \cap (AgVar \cup NVar \cup KVar)$.
$NewVar_p = Sub(\tau_p) \cap (AgVar \cup NVar \cup KVar) \setminus (NewVar_1 \cup ... \cup NewVar_{p-1})$ for $p > 1$.
Let $j > 0$ and suppose that the first $(j-1)$ messages are realizable by A and $Kn_{A,j-1}$ is defined, then:

- If in message number j, A is receiver, this message can be realized by A since A is passive in this action.
- If message number j is of the form (A, B_j, τ_j), this message can be realized by A if and only if: $\tau_j \in Synth_A(Kn_{A,j-1} \cup NewVar_{p_j})$ where p_j is the index of the message (A, B_j, τ_j) in P.

In both cases, we have : $Kn_{A,j} = \{B_j\} \cup Anal^*_A(\{\tau_j\} \cup Kn_{A,j-1})$.
A session template Ses_A is *realizable* if all its t-messages in this session are realizable by role A.
A protocol is *realizable* if all the session templates of all roles of the protocol are realizable. Clearly, the above procedure is effective so one can decide whether a protocol is realizable.
For example on the TMN protocol with public key of the server, the evolution of the knowledge for each role is :

By now, we will consider only realizable protocols.

2.3 States. Transitions

We formulate now the semantics of a protocol as an infinite transition system where a state contains the set of current partial sessions of agents (it is actually

	A	B	S
Initial	$S, \mathcal{K}_{pub}(S)$	$S, \mathcal{K}_{pub}(S)$	
Step 1	$B, \mathcal{K}_a$		$A, B, \mathcal{K}_a$
Step 2		A	
Step 3		$\mathcal{K}_b$	$\mathcal{K}_b$
Step 4	$\mathcal{K}_b$		

a multiset because the same agent may have several "identical" partial sessions at the same time) and a transition corresponds to a send or a receive event. As in [Low95] we assume that every message is intercepted by the intruder, so w.l.o.g. one consider that every sent message is sent to the intruder, and every received message is received from the intruder, so we have two types of events the *send* and *receive* ones.

States A *valuation* v of a set of component template $\mathcal{T}$ is a function that associates to each term $\tau \in \mathcal{T}$ a concrete term $\bar{\tau} = v(\tau)$, the value of which is in $Value$ (i.e. to each component template is associated its value). We consider here constants as variables for which the valuation is fixed.
Let $(\tau_j)_{j=1,\ldots,k}$ be the list of contents of the t-messages of the session of a role A for a protocol P. Let v_j be a valuation for $Kn_{A,j}$. We denote $\tau_j[v_j]$ the concrete term we obtain when substituting in term τ_j to each maximal subterm τ' which is in $Kn_{A,j}$ the value $v_j(\tau')$. One can remark that τ_j is built in a unique way from its maximal subterms which are in $Kn_{A,j}$.
A *partial session* (or simply session) σ is determined by its length l, a role A and a valuation $v_l(A)$ for the knowledge $Kn_{A,l}$. The role of session σ will be denoted R_σ. The role A, the length l and the valuation v_l permit to define the list of the l first messages received by the agent playing this role in this session. It is the list of concrete messages $(\tau_j[v])_{j=1,\ldots,p\leq k}$, where $(\tau_j)_{j=1,\ldots,k}$ is the list of the t-messages of the session of role A.
A *state* is a multiset of partial sessions like in [CDL+99].

Transitions The formalization of the evolution of the state of the system via receive or send events is the most delicate part of the modeling. An *admissible state* is a state reachable from the initial state using transitions labeled by the following events:

- *send event* : tuple $(a, \longrightarrow, (a, b, m))$ where a, b are agents and (a, b, m) is a concrete message. It corresponds to the event "agent a sends (intentionally to agent b) the message m and this message is received by the intruder".
- *receive event* : tuple $(a, \longleftarrow, (a, b, m))$ where a, b are agents and (a, b, m) is a concrete message. It corresponds to the event "The intruder sends to agent b a message m and agent b believes that this message has been sent by agent a".

The *knowledge of the intruder* denoted $IntrKn$, is the set of values known by the intruder and that he cannot decompose more. It will be described more precisely

below.
• Send transitions
We have a transition from state S to state S' labeled by the send event $(a, \rightarrow, (a, b, m))$, denoted by $S \overset{(a,\rightarrow,(a,b,m))}{\longrightarrow} S'$ if the following conditions are satisfied:

1. a, $b \in Agent$.
2. There exists in S a partial session $\sigma = (A, v_l)$ of length l for which the next message is a send event or agent a starts a partial session for a role A in which the first message of Ses_A is a message sent by A.
3. $(a, b, m) = \tau_{l+1}[v_{l+1}]$ where v_{l+1} is a valuation defined as follows:
   (a) $(v_{l+1} \mid BasKn_{R_\sigma,l}) = (v_l \mid BasKn_{R_\sigma,l})$
   (b) $v_{l+1} \mid (BasKn_{R_\sigma,l+1} \setminus BasKn_{R_\sigma,l})$ must satisfy the rules
   – The values are of the correct type, i.e. values for nonces, agents and short term keys belong to the respective sets respectively $Nonce$, $Agent$, Key.
   – The valuation is injective on the set of nonces and the set of keys, and values are "fresh", i.e., if X is a variable for a nonce (resp. a key) belonging to $BasKn_{R_\sigma,l+1} \setminus BasKn_{R_\sigma,l}$, then $v_{l+1}(X)$ is not in the set of valuations of nonce variables (resp. key variables) for all the partial sessions of state S.
   – $CrKn_{R_\sigma,l+1} = CrKn_{R_\sigma,l}$ and for coherence $(v_{l+1} \mid CrKn_{R_\sigma,l+1}) = (v_l \mid CrKn_{R_\sigma,l+1})$.
4. S' is the state we obtain when replacing one exemplary of session $\sigma = (l, A, v_l)$ by $\sigma' = (l+1, A, v_{l+1})$. (It corresponds to increasing the list of concrete messages of the partial session σ with the concrete message (a, b, m)).

The knowledge of the intruder $\mathcal{I}$ at state S' is :
$IntrKn_{S'} = Anal^*_{\mathcal{I}}(IntrKn_S \cup \{(a, b, m)\})$
• Receive transitions
We consider here only the receive events where the message is accepted by the receiver.
We have a transition from state S to state S' labeled by the receive event $(a, \leftarrow\ , (a, b, m))$, denoted by $S \overset{(a,\leftarrow,(b,a,m))}{\longrightarrow} S'$ if the following conditions are satisfied:

1. a, $b \in Agent$.
2. There exists in S a partial session $\sigma = (l, A, v_l)$ for which the next message is a receive event, or (case $l = 0$) agent a starts a partial session for a role A in which the first message of Ses_A is a message received by A.
3. $(b, a, m) = \tau_{l+1}[v_{l+1}]$ where v_{l+1} is a valuation defined as follows:
   (a) $v_{l+1} \mid BasKn_{R_\sigma,l} = v_l \mid BasKn_{R_\sigma,l}$
   (b) $v_{l+1} \mid (BasKn_{R_\sigma,l+1} \setminus BasKn_{R_\sigma,l})$ must satisfy the rules
   – values belong to the set $Synth_{\mathcal{I}}(S)$ defined above.
   – values of agent variables belong to $Agent$.
   (c) $v_{l+1} \mid (CrKn_{R_\sigma,l+1} \cap CrKn_{R_\sigma,l}) = v_l \mid (CrKn_{R_\sigma,l+1} \cap CrKn_{R_\sigma,l})$.

   (d) $v_{l+1} \mid (CrKn_{R_\sigma,l+1} \setminus CrKn_{R_\sigma,l})$ has values in $Synth_{\mathcal{I}}(S)$.
4. S' is the state we obtain when replacing an exemplary of partial session σ with $\sigma' = (l+1, A, v_{l+1})$. (It corresponds to increasing the list of concrete messages of the partial session σ with the concrete message (b, a, m).

The knowledge of the intruder $\mathcal{I}$ at state S' is :
$IntrKn_{S'} = Anal^*_{\mathcal{I}}(IntrKn_S \cup \{m\})$
The set $Synth_{\mathcal{I}}(S)$ is the set of concrete terms that the intruder can build at state S. It is the least set containing $IntrKn_S$ and satisfying:

– $\tau_1, ..., \tau_p \in Synth_{\mathcal{I}}(S) \Rightarrow < \tau_1, ..., \tau_p > \in Synth_{\mathcal{I}}(S)$.
– $Agent \subset Synth_{\mathcal{I}}(S)$.
– For every agent a, the long term key $\mathcal{K}(a, \mathcal{I})$ is in $Synth_{\mathcal{I}}(S)$.
– For every agent a, the long term key $\mathcal{K}_{pub}(a)$ is in $Synth_{\mathcal{I}}(S)$.
– For every term $\tau \in Synth_{\mathcal{I}}(S)$ and for every key $\mathcal{K} \in (IntrKn_S \cap Key)$, $\{\tau\}_{\mathcal{K}} \in Synth_{\mathcal{I}}(S)$.

A *trace* of a protocol is a sequence $S_0 \xrightarrow{e_1} S_1 \xrightarrow{e_2} ...S_{n-1} \xrightarrow{e_n} S_n$ where S_0 is the initial state and each $S_{i-1} \xrightarrow{e_i} S_i$ is a transition.
The initial knowledge of the intruder $IntrKn_{S_0}$ is given by the protocol.

Remark. One can notice that the rules applied by an honest agent in order to accept a message correspond to a very weak control of the message. The agent makes only *equality* tests, it has no possibility to control for example the depth of encryption, the correct type of values and so on.

2.4 Secrecy

In the literature, generally the definitions of secrecy are very dependent on the chosen model and restrictive, i.e. sufficient for the hypothesis made by the authors but not applicable in a more general context. The definition we give here seems very general, at least as far as the concern is the secrecy of values and not of properties.

Definition 1. *The* secret *of a variable X for a nonce or a short term key can be broken from the point of view of A if there exists a reachable state S containing a partial session σ of length l for role A with valuation v_l for $Kn_{A,l}$ such that*

1. *$BasKn_{A,l}$ contains X and the set $\mathcal{R}$ of roles of the protocol*
2. *$\mathcal{I}$ does not belong to the valuation $v_l(\mathcal{R})$ ($\mathcal{I}$ does not participate to the partial session σ from the point of view of A)*
3. *$v_l(X) \in IntrKn_S$.*

As one can observe, the notion of secrecy implies two parameters: a *variable* for which the secret is broken and a *role* which can claim the fact. We have to justify points 1 and 2. Why should the set $\mathcal{R}$ be in $BasKn_{A,l}$? Because as far as the agent involved in the partial session σ does not know all its partners in this session, it cannot claim whether it is correct that the agent $\mathcal{I}$ knows the value

$v_l(X)$. Indeed, if $\mathcal{I}$ participates in an honest way to the session it is normal that $v_l(X) \in IntrKn_S$. For the same reason the condition that $\mathcal{I}$ does not belong to the valuation $v_l(\mathcal{R})$ is required. An unsolved question is how to define secrecy in the case when the set of roles does not belong to the knowledge of each role at the end of its partial session.
There is a well-known attack [LR97] on the protocol TMN of Example 1. An intruder $\mathcal{I}_A$ acts as if it was A:
$01 - \mathcal{I}_a \rightarrow S :< b, \{K_i\}_{\mathcal{K}_{pub}(S)} >$
$02 - S \rightarrow b :< a, b >$
$03 - b \rightarrow S :< a, \{K_b\}_{\mathcal{K}_{pub}(S)} >$
$04 - S \rightarrow \mathcal{I}_a :< B, \{K_b\}_{K_i} >$
In this attack, the secret is broken for the variable K_b from the point of view of B because the trace given here reaches a state containing a partial session for role B satisfying the above three conditions.

Given a protocol, the variables which can be learnt by an external observer of the protocol are called *revealed variables*. The others (those which remain unaccessible to this observer) are called *unrevealed variables*.
More precisely, given a protocol $P = (A_i, B_i, M_i)_{i=1,...,k}$, a variable X for a nonce or a key is revealed in P if $X \in Anal_C^*(\{M_1, ..., M_k\})$ for some C not being a role of P. The set of revealed variables of a protocol is clearly computable. In an obvious way, the secret can be broken for every revealed variable from the point of view of every role. Thus, the interesting question is "can the secret be broken for an unrevealed variable". The next section answers to this question by giving a sufficient condition which guarantees that the protocol preserves the secrecy of unrevealed variables for nonces and short term key variables.

3 A sufficient condition for secrecy

3.1 Well-Composed Protocol

A *signature* of a protocol is constituted by a nonce variable which is called the session nonce and a fixed list of the agent roles $< n, A_1, ..., A_p >$.

Definition 2. *A protocol is* well composed *if :*

1. *Encryption is of depth at most two.*
2. *Private long term asymmetric and long term symmetric keys are never transmitted.*
3. *There exists a signature $\mathcal{S}$ such that*
   - *the content of every t-message is a term of the form : $< \mathcal{S}, \{\mathcal{S}, m\}_{\mathcal{K}_{priv}(A)} >$ where A is the sender of the message,*
   - *every subterm of the protocol which is an encrypted term has the form $\{< \mathcal{S}, ... >\}_{\mathcal{K}}$ (it contains the signature on the left inside the encryption).*
4. *Two different encrypted terms which are encrypted by the same type of keys (public, private, ...) must have a different number of elements. More precisely, if $\{< \tau_1, ...\tau_k >\}_{\mathcal{K}}$ and $\{< \tau'_1, ..., \tau'_{k'} >\}_{\mathcal{K}'}$ are two different subterms of a protocol P and $\mathcal{K}, \mathcal{K}'$ are of the same type, then $\mathcal{K} \neq \mathcal{K}'$.*

Let us comment the four given conditions. Condition (4) helps to prevent the intruder from passing off a term $\{\tau\}_{\mathcal{K}}$ as a term $\{\tau'\}_{\mathcal{K}'}$ while these terms are intended to be distinct terms in the specification. Another way to obtain the same effect would be to use tagging as it is done in several papers [BP03,HLS00,RS03]. In these papers, tagging is used to prove decidability of secrecy for tagged protocols, but it is not a sufficient condition for secrecy. Condition (3) is reasonable and permits to know at each moment who is supposed to be implied in the session. An attack on TMN protocol is due to the fact that this condition is not satisfied. Condition (2) is always recommended [AN96]. At last, condition (1) is not essential here. We are convinced that this hypothesis could be relaxed, but it would make the proof more complicated.
The TMN protocol is not well composed. Here is a modified version which is well composed:
$01 - A \rightarrow S : \mathcal{S}, \{\mathcal{S}, B, \{\mathcal{S}, K_a\}_{\mathcal{K}_{pub}(S)}\}_{K_{priv}(A)}$
$02 - S \rightarrow B : \mathcal{S}, \{\mathcal{S}, B, A\}_{K_{priv}(S)}$
$03 - B \rightarrow S : \mathcal{S}, \{\mathcal{S}, A, \{\mathcal{S}, K_b\}_{\mathcal{K}_{pub}(S)}\}_{K_{priv}(B)}$
$04 - S \rightarrow A : \mathcal{S}, \{\mathcal{S}, B, \{\mathcal{S}, K_b\}_{K_a}\}_{K_{priv}(S)}$

The attack presented in the previous section fails in this new version because the intruder cannot impersonate A at the first step of the attack.

Theorem 1. *A well composed protocol preserves the secrecy of unrevealed variables for nonces and short term key variables.*

Before giving the proof of this theorem let us recall the sufficient condition given in [Bea04] to preserve secrecy in case of symmetric encryption, and show with a counter example that this condition is not enough for protocols involving asymmetric encryption. This sufficient condition was:

1. Encryption is of depth one.
2. Long term keys are never transmitted.
3. There exists a signature $\mathcal{S}$ such that every subterm of the protocol which is an encrypted term has the form $\{< \mathcal{S}, ... >\}_{\mathcal{K}}$ (it contains the signature on the left inside the encryption).

Here is a variant of TMN protocol which satisfies this condition.
Let $\mathcal{S} =< N, A, B >$ where N is a nonce.

$01 - A \rightarrow S : \mathcal{S}, B, \{\mathcal{S}, K_a\}_{\mathcal{K}_{pub}(S)}$
$02 - S \rightarrow B : \mathcal{S}, B, A$
$03 - B \rightarrow S : \mathcal{S}, A, \{\mathcal{S}, K_b\}_{\mathcal{K}_{pub}(S)}$
$04 - S \rightarrow A : \mathcal{S}, B, \{\mathcal{S}, K_b\}_{K_a}$.

Clearly, an attack similar to the one given before can be repeated.
The next proposition expresses the fact that a well composed protocol guarantees some authenticity: if an agent a receives in a partial session where it plays role A a message m from another agent b and a thinks that b plays role B and that m corresponds to the message number i of the protocol, indeed b has sent this message for this purpose.

Proposition 1. *Let r be a trace of a well composed protocol. If r contains a transition $S \overset{(a,\leftarrow,(b,a,\tau))}{\longrightarrow} S'$ where S contains a partial session σ of length l belonging to an agent a for the role A, and σ is replaced in S' by a partial session σ' of length $l+1$ where b has role B, then there is a previous transition in r of the form $S_1 \overset{(b,\rightarrow,(b,a,\tau))}{\longrightarrow} S_1'$ where S contains a partial session σ_1 of length l_1 belonging to agent b for the role B and σ_1 is replaced in S_1' by a partial session σ_1' of length l_1+1 where the message number $l+1$ of role A is exactly the message number l_1+1 of role B.*

Proof. If a accepts the message, it means that the message is of the right form, namely : (b,a,τ) with $\tau =< s_1, \{s_1, \tau'\}_{\mathcal{K}_{priv}(b)} >$.
Actually τ must be encrypted by $\mathcal{K}_{priv}(b)$ since it is supposed to have been sent by b. Moreover, a controls that the signature located in the first elements of τ is the same as the signature contained at the beginning of the encrypted element. As a consequence, b is the agent who encrypted τ. Due to the last condition of the definition of a well composed protocol, a also controls that the number of elements in τ corresponds to the number of elements awaited by a in this session, so necessarily, b built τ to send a message number $l+1$ for the role A, and this role is played by a because a has in the signature the place corresponding to role A.

We now translate in an equivalent form the property of secrecy for a well composed protocol. Let r be a run with a length l, X be an unrevealed variable for a nonce or a short key, x be a value, T be a time less than or equal to l, and t be a positive integer. The tuple (r,X,x,T,t) satisfies $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) iff:

- $\mathcal{P}_1$: in r at some time $T' < T$, in one of its partial sessions whose signature does not contain $\mathcal{I}$, an honest agent a generates the value x to assign to the unrevealed variable X and at time T, $\mathcal{I}$ learns the value x in t steps.
- $\mathcal{P}_2$: in r, in one of its partial sessions whose signature does not contain $\mathcal{I}$, an honest agent a learns the value x of the unrevealed variable X at time T in t steps and at the end of the run r the value x belongs to the knowledge of $\mathcal{I}$, i.e. $x \in IntrKn_l$. Moreover, there is no tuple of the form (r,X',x,T',t') satisfying $\mathcal{P}_1$, in other words x is not a value generated by an honest agent to assign to an unrevealed variable.

Lemma 1. *A well composed protocol preserves the secrecy of unrevealed variables for nonces and short term key variables from the point of view of every role iff there does not exist an unrevealed variable X for a nonce or a short term key, a value x, a run r with length l, a time $T \leq l$ and a positive integer t such that the tuple (r,X,x,T,t) satisfies $\mathcal{P}_1 \vee \mathcal{P}_2$.*

Proof. Firstly assume that there exists an unrevealed variable X for a nonce or a short term key, a value x, a run r with length l, a time $T \leq l$ and a positive integer t such that the tuple (r,X,x,T,t) satisfies $\mathcal{P}_1 \vee \mathcal{P}_2$.

If (r,X,x,T,t) satisfies $\mathcal{P}_1$ then in r at some time $T' < T$, in some partial session σ for a role A, with a signature that does not contain $\mathcal{I}$, an honest agent

a generates the value x to assign to the unrevealed variable X and at time T,in some state S, $\mathcal{I}$ learns the value x in t steps. Clearly the secret of variable X can be broken from the point of view of role A. Actually in state S, the extension of partial session σ has a length l and a valuation v_l for $Kn_{A,l}$ such that $BasKn_{A,l}$ contains X and the set $\mathcal{R}$ of roles of the protocol, $\mathcal{I}$ does not belong to the valuation $v_l(\mathcal{R})$ and $v_l(X) \in IntrKn_S$.

If (r, X, x, T, t) satisfies $\mathcal{P}_2$, in the same way let A be the role played by a in its partial session. The secret of variable X is broken from the point of view of role A. The "if" part of the Lemma is proved.

Secondly assume that in a well composed protocol, the secret of a variable X can be broken from the point of view of a role A. It means there exists a reachable state S containing a partial session σ of length l' for role A with valuation $v_{l'}$ for $Kn_{A,l'}$ such that $BasKn_{A,l'}$ contains X and the set $\mathcal{R}$ of roles of the protocol, $\mathcal{I}$ does not belong to the valuation $v_{l'}(\mathcal{R})$ and $v_{l'}(X) \in IntrKn_S$. Let r be a run from the initial state of the protocol to state S, let l be its length and let $x = v_{l'}(X)$. Since $BasKn_{A,l'}$ contains X it means that at some moment in the partial session σ the agent $a = v_{l'}(A)$ either generates the value x to assign to the variable X (first case) or a learns it (second case).

In the first case, let T' be the moment when a generates the value x. Since $v_{l'}(X) \in IntrKn_S$, there is a time $T > T'$ when the intruder learns x in t steps, more precisely, if S_i denotes the i-th state of run r, there is a state S_T such that $x \in IntrKn_{S_T}$ and $x \notin IntrKn_{S_{T-1}}$. In this first case the tuple (r, X, x, T, t) satisfies $\mathcal{P}_1$.

In the second case, in the partial session σ, a has not generated x (may be a has generated x in another session) and a has learnt the value x of X at time $T \leq l$ in t steps. If there exists a tuple (r, X', x, T'', t') satisfying $\mathcal{P}_1$, we are done. If not, then the tuple (r, X, x, T, t) satisfies $\mathcal{P}_2$. The "only if" part of the Lemma is proved.

Well composed protocols have an invariant property which is stated below not very formally:

Lemma 2. *If in a trace r, at time T_1, an honest agent a generates a value x to substitute to an unrevealed variable X in a message m that he sends with a signature $\mathcal{S}$ not including $\mathcal{I}$, then, as long as $\mathcal{I}$ does not learn x, x has only occurrences in encrypted components $\tau = \{\mathcal{S}, \ldots, x, \ldots\}_K$ where the term τ has been encrypted by an honest agent belonging to $\mathcal{S}$ and put by this same agent in a message m' in which the place where is x inside τ is the place of X.*

Proof. The property is true at t_1. Let $t > t_1$ and assume $\mathcal{I}$ does not know x at t. If x is in an encrypted component, this one has been encrypted by an honest agent b, in some session otherwise, $\mathcal{I}$ knows x. The value x is by recurrence hypothesis for b the value of an unrevealed variable X, and then in the component encrypted by b to send in a message m', x is in place of X.

Proposition 2. *In a well composed protocol, there does not exist any tuple satisfying $\mathcal{P}_1 \vee \mathcal{P}_2$.*

Proof. Suppose there exist tuples (r, X, x, T, t) for which $\mathcal{P}_1 \vee \mathcal{P}_2$ holds. Consider the total strict order relation: $(r, X, x, T, t) < (r', X', x', T', t')$ iff $T < T'$ or $(T = T'$ and $t < t')$ and take a minimal tuple (r, X, x, T, t) satisfying $\mathcal{P}_1 \vee \mathcal{P}_2$. Let us examine the two cases :

- The tuple (r, X, x, T, t) satisfies $\mathcal{P}_1$.
  Let $\mathcal{S}$ be the signature not including $\mathcal{I}$ of the partial session in which at time $T_1 < T$, the honest agent a generates the value x to substitute to the unrevealed variable X in a message m. Let τ be the concrete term from which $\mathcal{I}$ learns x at time T in t steps. We consider here the term of the very last operation of decryption made by $\mathcal{I}$ to learn x. This term τ has a value of type *Cypher* and it is of the form $\{..., x, ...\}_{\mathcal{K}}$. This term has not been built by $\mathcal{I}$ in r before, otherwise $\mathcal{I}$ would have known x before, and (r, X, x, T, t) would not be minimal. So it has been built by an honest agent d, and for this reason, due to Lemma 2, it has been built by an honest agent d belonging to $\mathcal{S}$ and put by this same agent in a message m' in which the place where is x inside τ is the place of X. Thus the term τ is of the form $\{\mathcal{S}, ..., x, ...\}_{\mathcal{K}}$, because the places of the unrevealed variable X cannot be the places of the components of $\mathcal{S}$. There are 3 cases for $\mathcal{K}$:
  1. $\mathcal{K}$ is a long term symmetric key
  2. $\mathcal{K}$ is a public long term key
  3. $\mathcal{K}$ is a short term symmetric key.

  Actually, $\mathcal{K}$ cannot be a private key, because this private key should be $\mathcal{K}_{priv}(d)$ and the component would not be in a unrevealed position. Let's go through each of the three cases :
  1. Since d has built the term τ, and since the signature inside τ does not contain $\mathcal{I}$, $\mathcal{K}$ is equal to some $\mathcal{K}(c, d)$ where c is a honest agent. So $\mathcal{I}$ cannot decrypt τ. This first case is not possible.
  2. The key $\mathcal{K}$ cannot be the public key of $\mathcal{I}$, because $\mathcal{I}$ is not in the signature $\mathcal{S}$. So $\mathcal{I}$ cannot decrypt τ. This second case is also impossible.
  3. In the partial session s where d builds the term τ at time $T' < T$, either d knows $\mathcal{K}$ or he generates it. In both cases in the message m' sent by d which contains τ, $\mathcal{K}$ is in place of an unrevealed variable Y, otherwise, x itself would be in an unrevealed place. Thus the secret is broken from the point of view of the role played by d in this partial session s and for the variable Y in the place of $\mathcal{K}$ in the term τ inside the message m'. As for $\mathcal{K}$ there are two possibilities. Either d has generated it or he has learnt it at time at most T' in in this session s. In the first case, there is a tuple $(r, Y, \mathcal{K}, T", t')$ which satisfies $\mathcal{P}_1$ with $T" < T$. In the second case there is a tuple $(r, Y, \mathcal{K}, T", t')$ which satisfies $\mathcal{P}_2$ with $T" < T$. It contradicts the minimality of (r, X, x, T, t).
- The tuple (r, X, x, T, t) satisfies $\mathcal{P}_2$.
  At time T, in t steps an honest agent a in a partial session s whose signature $\mathcal{S}$ does not contain $\mathcal{I}$ learns the value x. Let τ be the last encrypted concrete term from which a learns x. This term τ was contained in a message m_1 received by a at time T or before and x in this message m_1 is in place

of an unrevealed variable X. This message which has been accepted by a has the form $< \mathcal{S}, \{\mathcal{S}, ..., \tau, ...\}_{K_{priv}(a_1)} >$ where $a_1 \in \mathcal{S}$ or $< \mathcal{S}, \tau >$. Here we use the fact that the protocol has an encryption depth at most two. If m_1 was equal to $< \mathcal{S}, \tau >$, then we would have $\tau = \{.., x, .\}_{K_{priv}(c)}$, and x would be in place of a revealed variable. So the message has the form $< \mathcal{S}, \{\mathcal{S}, ..., \tau, ...\}_{K_{priv}(a_1)} >$. Moreover the term τ has the form $\{..., x, ...\}_{\mathcal{K}}$. Let us observe that, the number of components of $< \mathcal{S}, \{\mathcal{S}, ..., \tau, ...\}_{K_{priv}(a_1)} >$ permits to the agent a to identify the index i of the message. For the same reason since the term $\{\mathcal{S}, ..., \tau, ...\}_{K_{priv}(a_1)}$ has been built by a_1 in some partial session s_1, a_1 has built this term in order to send the message m_1, and the value of τ in the partial session s_1 of a_1 and in the partial session s of a are associated to the same symbolic term of the protocol. Let us come back to $\tau = \{..., x, ...\}_{\mathcal{K}}$.
If the agent a_1 in the partial session s_1 builds the term τ by encryption with $\mathcal{K}$ before sending the message m_1, it means that in this session s_1 at this moment, x is known by a_1. So in this session s_1, x is learnt by a_1 at a time $T' < T$ in t' steps. Indeed, x is not generated by a_1 at least in this session because for a_1, in this session, x is the value of X which is unrevealed, which would contradict the fact that (r, X, x, T, t) satisfies $\mathcal{P}_2$. Thus replacing a by a_1 we get a tuple (r, X, x, T', t') satisfying $\mathcal{P}_2$ which contradicts the minimality of (r, X, x, T, t). So the agent a_1 in the partial session s_1 does not build the term τ by encryption with $\mathcal{K}$ before sending the message m_1. It means that this term τ has been obtained from a previous message m_2 that the agent a_1 received in its partial session s_1. Thus we can iterate our reasoning for a_1 instead of a, but only a finite number of times because the run r is finite. Thus we get in any case a contradiction.

So we've proved that there cannot exist any tuple satisfying $\mathcal{P}_1 \vee \mathcal{P}_2$, which by induction, proves Proposition 2.

Theorem 1 is a direct consequence of Lemma 1 and Proposition 2.

A well composed version of TMN protocol would be:

Example 2. .
$01 - A \rightarrow S : \mathcal{S}, \{\mathcal{S}, B, \{\mathcal{S}, K_a\}_{\mathcal{K}_{pub}(S)}\}_{K_{priv}(A)}$
$02 - S \rightarrow B : \mathcal{S}, \{\mathcal{S}, B^2, A\}_{K_{priv}(S)}$
$03 - B \rightarrow S : \mathcal{S}, \{\mathcal{S}, B^2, A, \{\mathcal{S}, B, K_b\}_{\mathcal{K}_{pub}(S)}\}_{K_{priv}(B)}$
$04 - S \rightarrow A : \mathcal{S}, \{\mathcal{S}, B^5, \{\mathcal{S}, K_b\}_{K_a}\}_{K_{priv}(S)}$
(B^n means a sequence of n B)

The previous attack fails at the first step because the intruder cannot impersonate the agent a for role A in the first message of the protocol. There are other attacks on this protocol which use the algebraic properties of the XOR algorithm used for encryption, but it is out of the scope of our framework.

Remark. As noticed by M. Abadi in [Aba99], authenticity is dual to secrecy in the sense that authenticity concerns the source of the messages while secrecy concerns their destination. Nevertheless, it seems that it is hard to ensure secrecy without some phase of authentication. If we look at attacks that breach the secrecy without using specific algebraic properties of the encryption algorithms, very often the intruder exploits some weakness of the protocol with respect to authentication.

4 An algorithm for securing protocols

In this section we describe a very simple algorithm $\mathcal{A}$ which transforms a protocol P into a protocol $P' = \mathcal{A}(P)$ which is secure w.r.t. secrecy and such that P' preserves the "intended goal" of P for a large class C of protocols. Surely, we have to define what means "to preserve the intended goal". We first describe the class C, secondly we give the algorithm $\mathcal{A}$ which can be applied to every protocol in the class C, then we define an equivalence relation over the set of protocols in C. Finally we prove that for every protocol P in C the protocol $\mathcal{A}(P)$ is equivalent to P and is well composed, so, $\mathcal{A}(P)$ is secure w.r.t. secrecy.

4.1 The class C and the algorithm $\mathcal{A}$

Definition 3. *A protocol is in C if it satisfies the following conditions:*

- *Encryption is of depth at most two.*
- *If in a template message (A, B, τ) there is a subterm of τ with an encryption depth equal to two, then $\tau' = \{\tau''\}_{K_{priv}(A)}$.*
- *Private long term asymmetric and long term symmetric keys are never transmitted.*

A lot of protocols belong to the class C : ISO/IEC 11770-3 Key Transport Mechanisms (1,2,3,4,5,6), Helsinki Protocol, TMN with public key protocol, Blake-Wilson-Menezes Secure Key Transport Protocol, Needham-Schroeder Public Key Protocol X.509 one-pass, two-pass, three-pass authentication, . . .
The algorithm $\mathcal{A}$ is the following :

1. Introduce a new variable N for a session nonce, and define a signature $\mathcal{S} =< N, R_1, R_2, ..., R_k >$ where $R_1, R_2, ..., R_k$ are the roles of the protocol P.
2. Transform the content m of each template message (A, B, m) according to the type of m :
   * If m is a tuple of n elements $(n \geq 1)$ and none of them is encrypted by $K_{priv}(A)$, replace m with $\{m\}_{K_{priv}(A)}$.
   * If $m =< \tau_1, ..., \tau_n > (n \geq 1)$ and at least one of the τ_i is encrypted by $K_{priv}(A)$, replace m with $\{m'\}_{K_{priv}(A)}$ where m' is the term we get, replacing each term $\tau_i = \{\tau_i'\}_{K_{priv}(A)}$ with τ_i'. In other terms, the encryption with $K_{priv}(A)$ is done over the tuple instead of some of its elements.
3. Replace in each template message, each subterm of the form $\{\tau\}_K$ by the subterm $\{< \mathcal{S}, \tau >\}_K$. Notice that, by associativity we have $< \mathcal{S}, \tau >=< N, R_1, R_2, ..., R_k, \tau >$

4. Replace each content m with $< \mathcal{S}, m >$.
5. If several terms of the protocol encrypted by the same type of key namely long term public type, long term private type, long term symmetric type or short term symmetric type have the same number of elements, add inside the term, after the signature, occurrences of the last role in order to get different numbers of elements for all the encrypted terms of the same type.

The well composed protocol of Example 2 is obtained applying this algorithm to Example 1.
We now prove that the protocol P' one obtains applying the algorithm $\mathcal{A}$ to a protocol $P \in C$ is in some sense equivalent to P, i.e. the new knowledge of each role is essentially the same as before, at least from the point of view of the nonces and the session keys appearing in the protocol P.

Definition 4. *Let P be a protocol in the class C and $P' = \mathcal{A}(P)$. The protocol P' is said* weakly equivalent *to P if for each role R_i for each step j, $BasKn_{R_i,j}(P') = BasKn_{R_i,j}(P) \cup \{R_1, ..., R_n\} \cup \{n\}$ and $CrKn_{R_i,j}(P') = \sigma(CrKn_{R_i,j}(P))$ where the $\sigma(\{\tau\}_k) = \{< \mathcal{S}, \tau >\}_k$ for every term τ. (Notice that terms of $CrKn_{R_i,j}$ have an encryption depth equal to 1 for protocols in the class C).*

In other terms, at every step, the basic knowledge is only increased by the set of roles and the nonce which is added, and the encrypted knowledge is the same except that the signature in inserted in the encrypted term.

Theorem 2. *Let P be a protocol in the class C and $P' = \mathcal{A}(P)$. The protocol P' is* weakly equivalent *to P and is well composed.*

Proof. Recall that for every role A, $Kn_{A,j}(P) = \{B_j\} \cup Anal^*_A(\{\tau_j\} \cup Kn_{A,j-1})$ for role A if the j-th template message of his partial session is (A, B_j, τ_j) or (B_j, A, τ_j).

Let $(A, B_j, < \mathcal{S}, \{\mathcal{S}, \tau'_j\}_{K_{priv(A)}} >)$ resp. $(B_j, A, < \mathcal{S}, \{\mathcal{S}, \tau'_j\}_{K_{priv(B_j)}} >)$ be the corresponding message in P'. We have

$Kn_{A,j}(P') = \{B_j\} \cup Anal^*_A(< \mathcal{S}, \{\mathcal{S}, \tau'_j\}_{K_{priv(A)}} >) \cup Kn_{A,j-1}(P')$,

where τ' is obtained from τ essentially by adding the signature in every encrypted term. So by induction on j,

$Kn_{A,j}(P') = BasKn_{A,j}(P) \cup \{R_1, ..., R_n\} \cup \{N\} \cup \sigma(CrKn_{A,j}(P))$

where $\mathcal{S} =< N, R_1, ..., R_n >$.

Remark The condition concerning the number of elements inside encrypted terms of the same type can be obtained more simply by adding different integers inside the encrypted terms which permit to identify them. Proceeding in this way, the messages will be shorter.

5 Conclusion

We have given a simple sufficient condition to guarantee the secrecy for cryptographic protocols which use pairing and symmetric and/or asymmetric encryption. Secrecy is ensured for an unbounded number of agents, nonces, sessions,

without assuming any typing of terms. Moreover, for a large class of protocols we provide an algorithm which transforms a protocol into a secure one w.r.t. secrecy and preserves the "intended goal" of the original protocol. To our knowledge it is the first result of this type.
We have limited our work to protocols of depth at most two, which is reasonable from a practical point of view. It seems that we could get rid of this restriction easily, but the proof would be more technical. A drawback of our sufficient condition is that the systematic signature of messages with the private key of the sender increases the size of the message. It would be better to replace $< \mathcal{S}, \{\mathcal{S}, m\}_{K_{priv}(A)} >$ with $< \mathcal{S}, m, \{H(\mathcal{S}, m)\}_{K_{priv}(A)} >$ where H is a hash function. We propose to extend our study with more primitives, in particular with hash functions.

References

Aba99. Martín Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.

AC02a. R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *In Proc. CONCUR 02. Springer-Verlag, 2002.*, pages 499–514, 2002.

AC02b. Roberto M. Amadio and Witold Charatonik. On name generation and set-based analysis in the dolev-yao model. In *CONCUR*, pages 499–514, 2002.

AG00. M. Abadi and A.D. Gordon. A calculus for cryptographic protocols : The spi-calculus. In *Proceedings of the 27-th ACM Symposium on Principles of Programming Languages (POPL'00)*, pages 302–315, 2000.

AN96. M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. 22(1):6–15, 1996.

Bea04. Danièle Beauquier. Secrecy for cryptographic protocols is easy. In *Proceedings of SYNASC'04*, 2004.

Bla01. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings of the 14-th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society Press, June 2001.

BP03. Bruno Blanchet and Andreas Podelski. Verification of Cryptographic Protocols: Tagging Enforces Termination. In Andrew D. Gordon, editor, *Proceedings of FoSSaCS'03*, volume 2620 of *Lecture Notes in Computer Science*, pages 136–152, 2003.

CCM01. H. Comon, V. Cortier, and J. Mitchell. Tree automata with one memory, set constraints, and ping-pong protocols. In *Proc. of the Intern. Conf. on Automata, languagesand Programming (ICALP'01,)Lect. Notes in Comput. Sci., vol.2076*, pages 682–693, 2001.

CDL$^+$99. I. Cervesato, N. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proc. of the 12-th IEEE Computer Security Foundations Workshop*, 1999.

DLMS99. N. Durgin, P. Lincoln, J. C. Mitchell, and A Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols, Trento, Italy*, 1999.

DLMS04. N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(1):677–722, 2004.

DY83. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. on Information Theory*, 29(2):198–208, 1983.

HLS00. James Heather, Gavin Lowe, and Steve Schneider. How to Prevent Type Flaw Attacks on Security Protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW 13)*, pages 255–268, July 2000.

JG02. F.J. Thayer J.D. Guttman. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, 2002.

Low95. Gavin Lowe. An attack on the needham-schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, November 1995.

Low99. G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(2,3):89–146, 1999.

LR97. G. Lowe and A. W. Roscoe. Using csp to detect errors in the tmn protocol. *IEEE Transactions on Software Engineering*, 23(10):659–669, 1997.

Pau98. L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

RS03. R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proceedings of 23rd FST&TCS*, number 2914 in Lecture Notes in Computer Science, pages 363–374, Mumbai, India, December 2003.

Sch97. S. Schneider. Verifying authentication protocols with csp. In *Proceedings of the 10-th Computer Security Foundations Workshop (CSFW'97)*. IEEE Computer Society Press, 1997.

TMN90. M. Tatebayashi, N. Matsuzaki, and D.B. Newman. Key distribution protocol for digital mobile communication systems. In *Proceedings on Advances in cryptology: Proc. Crypto'89 Lect. Notes in Comput. Sci., vol.435*, pages 324–333, 1990.